

计算机组成原理

计算机组成原理课程组

(刘旭东、高小鹏、肖利民、牛建伟、栾钟治)

第六讲 MIPS处理器设计

- 一. 处理器设计概述
- 二. MIPS模型机
- 三. MIPS单周期处理器设计
 1. 单周期数据通路设计
 2. 单周期控制器设计
 3. 单周期性能分析
- 四. MIPS多周期处理器设计
 1. 多周期数据通路设计
 2. 多周期控制器设计
 3. 多周期性能分析
- 五. MIPS流水线处理器设计

4.1 MIPS 多周期数据通路设计

❖ 为什么不使用单周期实现方式

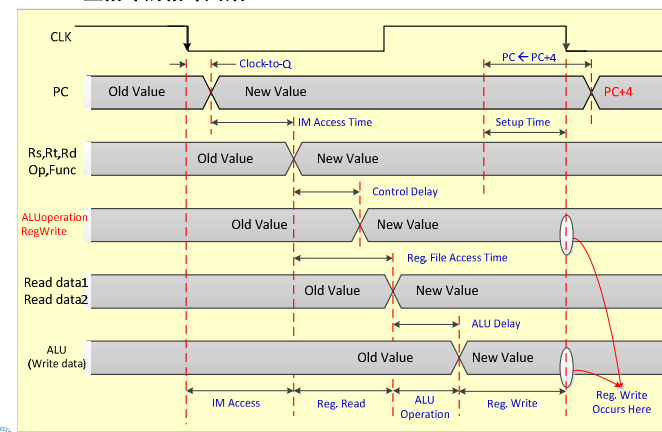
- 单周期设计中, 时钟周期对所有指令等长。而时钟周期由计算机中可能的最长路径决定, 一般为取数指令。但某些指令类型本来可以在更短时间内完成。

❖ 多周期方案

- 将指令执行分解为多个步骤, 每一步骤一个时钟周期, 则指令执行周期为多个时钟周期, 不同指令的指令周期包含时钟周期数不一样。
- 优点:
 - **提高性能**: 不同指令的执行占用不同的时钟周期数;
 - **降低成本**: 一个功能单元可以在一条指令执行过程中使用多次, 只要是在不同周期中(这种共享可减少所需的硬件数量)。

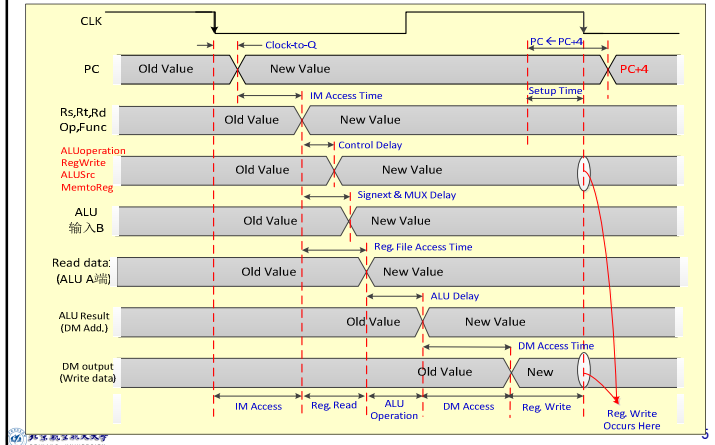
单周期数据通路分析

❖ R型指令的指令周期



单周期数据通路分析

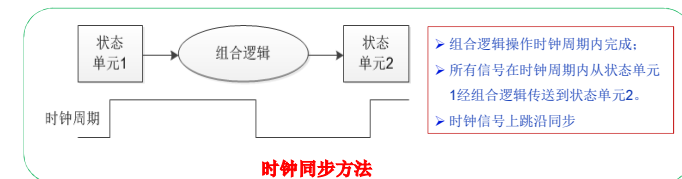
❖ LW指令的指令周期



4.1 MIPS多周期数据通路设计

❖ 多周期数据通路设计总体考虑

- **普林斯顿结构**：指令和数据使用同一个存储器；
- 一个ALU：R型指令算术逻辑运算、取指令后形成PC+4新值、及Beq指令转向地址计算 ($PC+4+Signext(imm16) < 2$)，都在ALU中完成；
- **时钟同步方法**：一个时钟周期内信号总是从一个状态单元经过组合逻辑处理后传送到另一个状态单元。所以指令每一步的执行总是从前一个状态单元接收输入，经过功能单元处理，在下一个时钟周期触发沿将结果写入下一个状态单元，因此数据通路中需要增加了一个或多个寄存器以保存指令各执行步骤形成的结果（输出值），以便在指令的后续时钟周期内继续使用。



4.1 MIPS多周期数据通路设计

1. R型指令多周期分析

❖ $R[r_d] \leftarrow R[r_s] \text{ op } R[r_t]$

| 步骤 | RTL描述 | 执行部件 | 备注 |
|------|--|------------|--|
| 取指令 | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ | 存储器 ALU | 增加一个寄存器保存指令: IR |
| 读寄存器 | $A \leftarrow R[IR[25:21]]$ $B \leftarrow R[IR[20:16]]$ | 寄存器堆 | 增加两个寄存器保存读取的数据R[rs]和R[rt]: A, B |
| 计算 | $ALUOut \leftarrow A \text{ op } B$ | ALU | 增加一个寄存器保存ALU结果: ALUOut |
| 写寄存器 | $R[IR[15:11]] \leftarrow ALUOut$ | 寄存器堆 | 完成寄存器数据写入 |

4.1 MIPS多周期数据通路设计

2. Lw指令多周期分析

❖ $R[r_t] \leftarrow M[R[r_s] + signext(imm16)]$

| 步骤 | RTL描述 | 执行部件 | 备注 |
|-------|---|------------|------------------------|
| 取指令 | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ | 存储器 ALU | 指令寄存器IR用于保存读取的指令 |
| 读寄存器 | $A \leftarrow R[IR[25:21]]$ | 寄存器堆 | 寄存器A保存从读取的数据R[rs] |
| 计算地址 | $ALUOut \leftarrow A + signext(IR[15:0])$ | ALU | ALUOut保存计算得到的内存地址 |
| 访问存储器 | $DR \leftarrow M[ALUOut]$ | 存储器 | 增加一个寄存器保存数据: DR |
| 写寄存器 | $R[IR[20:16]] \leftarrow DR$ | 寄存器堆 | 完成寄存器数据写入 |

4.1 MIPS多周期数据通路设计

3. Sw指令多周期分析

| Op (31-26) | Rs (25-21) | Rt (20-16) | 16 bit Address or Immediate (15-0) |
|---------------|---------------|---------------|---------------------------------------|
|---------------|---------------|---------------|---------------------------------------|

◆ $M[R[rs] + \text{signext}(imm16)] \leftarrow R[rt]$

| 步骤 | RTL描述 | 执行部件 | 备注 |
|-------|--|------------|-------------------|
| 取指令 | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ | 存储器 ALU | 指令寄存器IR用于保存读取的指令 |
| 读寄存器 | $A \leftarrow R[IR[25:21]]$ | 寄存器堆 | 寄存器A保存从读取的数据R[rs] |
| 计算地址 | $ALUOut \leftarrow A + \text{signext}(IR[15:0])$ | ALU | ALUOut保存计算得到的内存地址 |
| 访问存储器 | $M[ALUOut] \leftarrow R[IR[20:16]]$ | 存储器 | 把数据R[rt]写入存储器 |

4.1 MIPS多周期数据通路设计

4. Beq指令多周期分析

| Op (31-26) | Rs (25-21) | Rt (20-16) | 16 bit Address or Immediate (15-0) |
|---------------|---------------|---------------|---------------------------------------|
|---------------|---------------|---------------|---------------------------------------|

◆ If $((R[rs] - R[rt]) == 0)$ then $PC \leftarrow PC + 4 + \text{signext}(imm16) < 2$
else $PC \leftarrow PC + 4$

◆ 问题: ALU 要完成 $PC + 4$, $PC + 4 + \text{signext}(imm16) < 2$ 和 $R[rs] - R[rt]$
三次运算操作, 需要在三个不同时钟周期内实现, 如何合理安排?

| 步骤 | RTL描述 | 执行部件 | 备注 |
|---------------|---|-------------|------------------------------|
| 取指令 | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ | 存储器 ALU | 第一周期后 PC+4 完成, PC 为新值 |
| 读寄存器 / 计算转向地址 | $A \leftarrow R[IR[25:21]]$ $B \leftarrow R[IR[20:16]]$ $ALUOut \leftarrow PC + \text{Signext}[IR[15:0]] < 2$ | 寄存器堆 ALU | ALU 计算转向目标地址保存于 ALUOut |
| 完成转移 | if $(A - B == 0)$ then $PC \leftarrow ALUOut$ | ALU | ALU 减法结果不写入 ALU, PC 修改是一条件操作 |

4.1 MIPS多周期数据通路设计

◆ 指令数据通路综合 (R型指令、Lw、Sw、Beq)

| 步骤 | R型指令 | Lw指令 | Sw指令 | Beq指令 |
|-------------|---|--|--------------------------|--|
| 取指令 | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ ALU 完成加法结果送 PC | | | |
| 读寄存器 / 译码 | $A \leftarrow R[IR[25:21]]$ $B \leftarrow R[IR[20:16]]$ $ALUOut \leftarrow PC + \text{Signext}[IR[15:0]] < 2$ | | | |
| 计算 | $ALUOut \leftarrow A \text{ op } B$ | $ALUOut \leftarrow A + \text{Signext}(IR[15:0])$ | | If $(A - B == 0)$ then $PC \leftarrow ALUOut$ |
| R型完成 / 访问内存 | $R[IR[15:11]] \leftarrow ALUOut$ | $DR \leftarrow M[ALUOut]$ | $M[ALUOut] \leftarrow B$ | |
| 写寄存器 | | $R[IR[20:16]] \leftarrow DR$ | | |

4.1 MIPS多周期数据通路设计

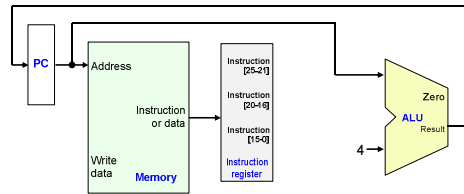
◆ 指令数据通路综合 (R型指令、Lw、Sw、Beq)

| 步骤 | R型指令 | Lw指令 | Sw指令 | Beq指令 |
|-------------|---|--|--|--|
| 取指令 | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ P0 | | | |
| 读寄存器 / 译码 | $A \leftarrow R[IR[25:21]]$ $B \leftarrow R[IR[20:16]]$ $ALUOut \leftarrow PC + \text{Signext}[IR[15:0]] < 2$ P1 | | | |
| 计算 | $ALUOut \leftarrow A \text{ op } B$ P6 | $ALUOut \leftarrow A + \text{Signext}(IR[15:0])$ P2 | | If $(A - B == 0)$ then $PC \leftarrow ALUOut$ P8 |
| R型完成 / 访问内存 | $R[IR[15:11]] \leftarrow ALUOut$ P7 | $DR \leftarrow M[ALUOut]$ P3 | $M[ALUOut] \leftarrow B$ P5 | |
| 写寄存器 | | $R[IR[20:16]] \leftarrow DR$ P4 | | |

4.1 MIPS多周期数据通路设计

1. 多周期数据通路构建 (P0)

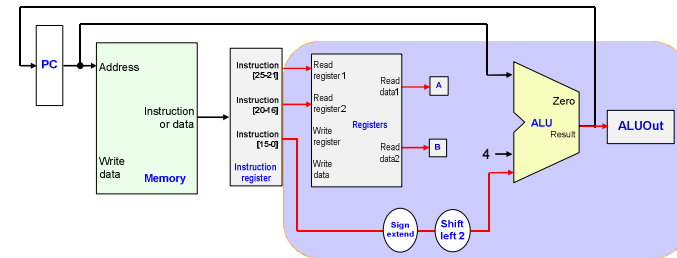
- ◆ $IR \leftarrow M[PC]$
- ◆ $PC \leftarrow PC + 4$



4.1 MIPS多周期数据通路设计

2. 多周期数据通路构建 (P1)

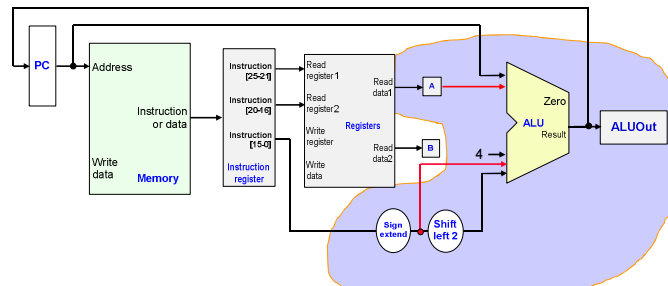
- ◆ $A \leftarrow R[IR[25:21]], B \leftarrow R[IR[20:16]]$
- ◆ $ALUOut \leftarrow PC + Signext[IR[15:0]] \ll 2$



4.1 MIPS多周期数据通路设计

3. 多周期数据通路构建 (P2)

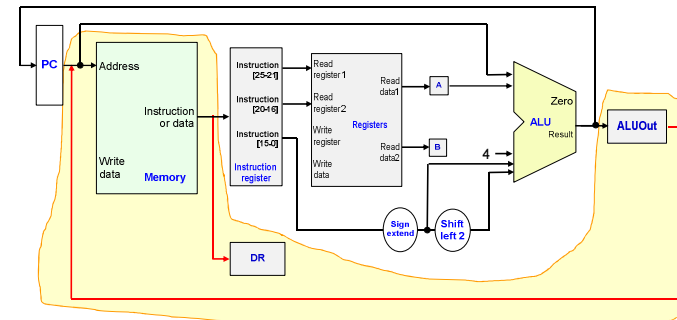
- ◆ $ALUOut \leftarrow A + Signext[IR[15:0]]$



4.1 MIPS多周期数据通路设计

4. 多周期数据通路构建 (P3)

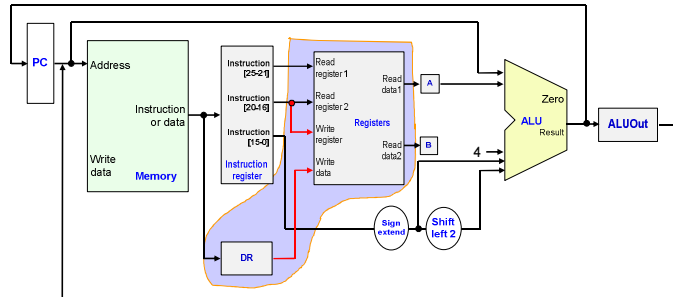
- ◆ $DR \leftarrow M[ALUOut]$



4.1 MIPS多周期数据通路设计

5. 多周期数据通路构建 (P4)

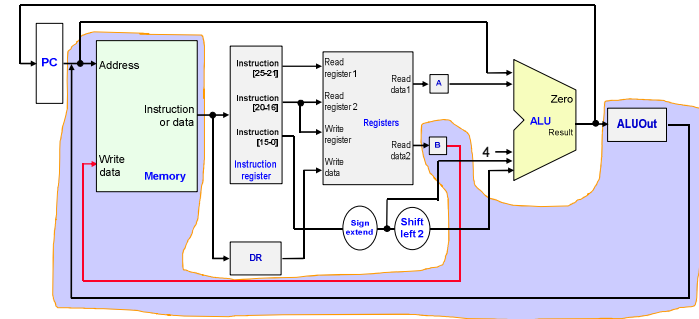
◆ $R[IR[20:16]] \leftarrow DR$



4.1 MIPS多周期数据通路设计

6. 多周期数据通路构建 (P5)

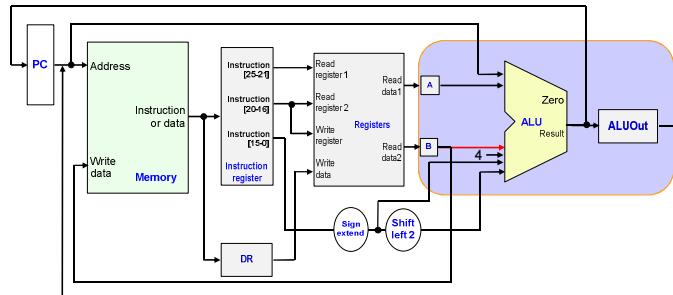
◆ $M[ALUOut] \leftarrow B$



4.1 MIPS多周期数据通路设计

7. 多周期数据通路构建 (P6)

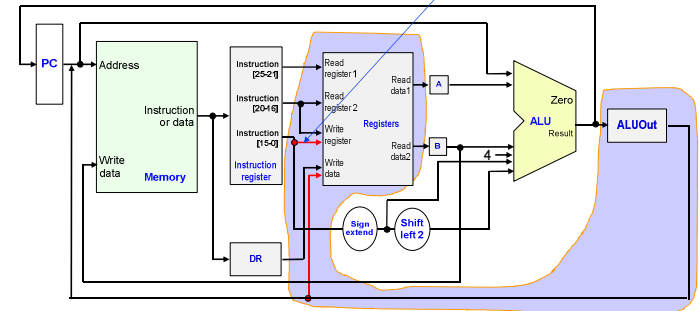
◆ $AUOut \leftarrow A \text{ op } B$



4.1 MIPS多周期数据通路设计

8. 多周期数据通路构建 (P7)

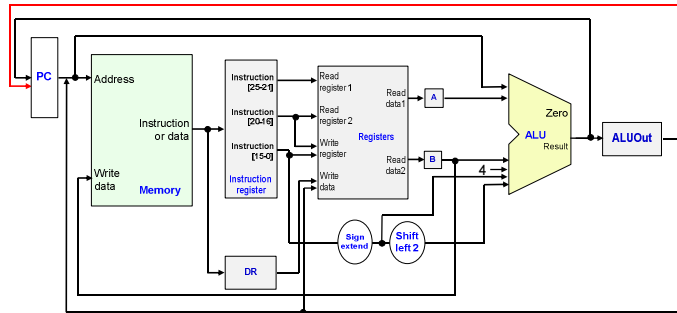
◆ $R[IR[15:11]] \leftarrow AUOut$



4.1 MIPS多周期数据通路设计

9. 多周期数据通路构建 (P8)

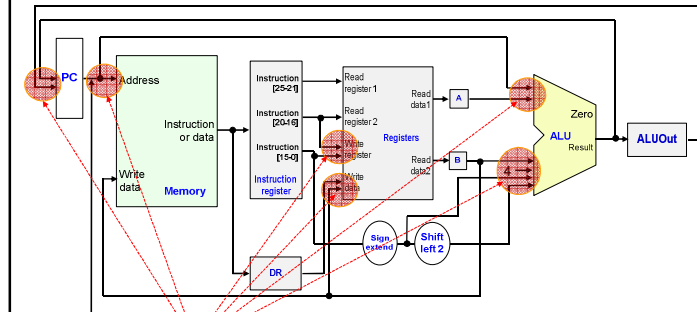
- ◆ If (A-B==0) then PC ← ALUOut



4.1 MIPS多周期数据通路设计

10. 多周期数据通路构建

- ◆ 存在多个输入源的，需要增加多路选择器MUX

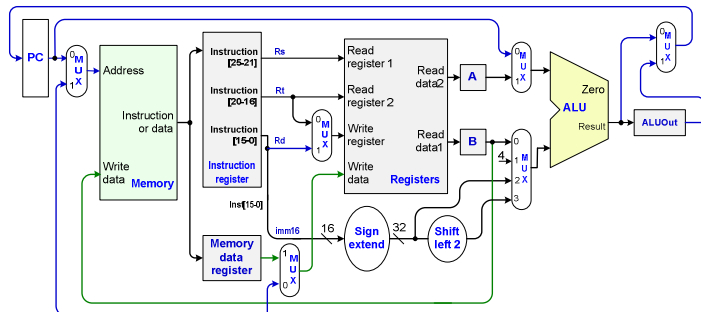


加入多路选择MUX

4.1 MIPS多周期数据通路实现

◆ MIPS的多周期数据通路

- PC: 输入端多路选择 (2选1)
- 存储器: 地址输入端多路选择 (2选1)
- ALU: 输入A端多路选择 (2选1), 输入B端多路选择 (4选1)
- 寄存器堆: 写地址输入端 (2选1), 写数据输入端 (2选1)



4.1 MIPS多周期数据通路设计

■ 增加J指令 (转移) 通路

- $PC \leftarrow PC + 4[31:28] \parallel add26 < 2$

| 步骤 | R型指令 | Lw指令 | Sw指令 | Beq指令 | J指令 |
|-----------|--|--------------------------------|---------------------------------|---------------------------------|-----|
| 取指令 | IR ← M[PC] PC ← PC + 4 | | | | |
| 读寄存器/译码 | A ← R[IR[25:21]] B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]] << 2 | | | | |
| 计算 | ALUOut ← A op B | ALUOut ← A + Signext[IR[15:0]] | If (A-B==0) then PC ← ALUOut | PC ← PC[31:28] IR[25:0] << 2 | |
| R型完成/访问内存 | R[IR[15:11]] ← ALUOut | DR ← M[ALUOut] | M[ALUOut] ← B | | |
| 写寄存器 | | R[IR[20:16]] ← DR | | | |

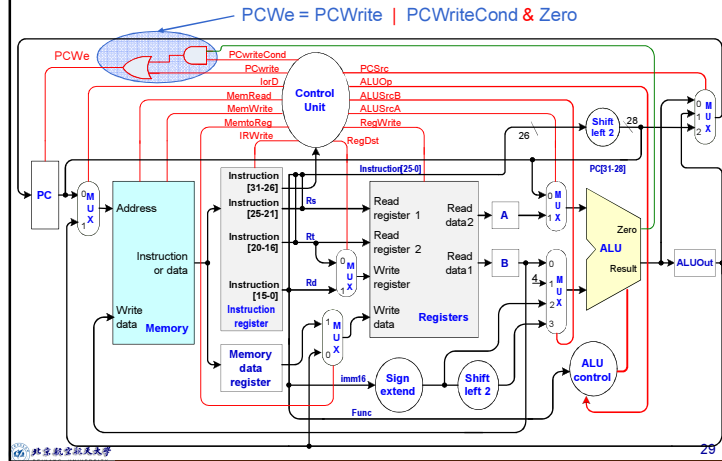
► 红色线路部分



五. MIPS流水线处理器设计

 北京航空航天大学

4.2 多周期控制器设计——完整数据通路与控制信号



4.2 多周期控制器设计

多周期通路控制信号

| 控制信号 | 失效时作用 | 有效时作用 |
|-------------|-------------------|---|
| RegDst | 寄存器堆写入端地址选择Rt字段 | 寄存器堆写入端地址选择Rd字段 |
| RegWrite | 无 | 把数据写入寄存器堆中对应寄存器 |
| ALUSrcA | ALU输入A端选择PC | ALU输入A端选择寄存器A |
| MemRead | 无 | 存储器读数据（输出） |
| MemWrite | 无 | 存储器写数据（输入） |
| MemtoReg | 寄存器堆写入端数据选择ALUOut | 寄存器堆写入端数据选择DR |
| IorD | 存储器地址输入选择PC | 存储器地址输入选择ALUOut |
| IRWrite | 无 | 存储器输出（指令）写入IR |
| PCWrite | 无 | PC写入，PC输入源由PCSrc选择 |
| PCWriteCond | 无 | 如ALU的Zero端输出有效，则PC写入，输入源由PCSrc选择（Beq指令） |

4.2 多周期控制器设计

多周期通路控制信号（续）

| 控制信号 | 取值（二进制） | 作用 |
|---------|---------|---|
| ALUOp | 00 | ALU执行加法 |
| | 01 | ALU执行加法 |
| | 10 | ALU操作由Func字段（IR[5:0]）决定 |
| ALUSrcB | 00 | ALU输入端B数据源选择寄存器B |
| | 01 | ALU输入端B数据源选择常数4 |
| | 10 | ALU输入端B选择符号扩展输出（Signext(imm16)） |
| | 11 | ALU输入端B选择移位器输出（Signext(imm16)<<2） |
| PCSrc | 00 | PC输入源选择ALU输出（取指阶段，PC+4） |
| | 01 | PC输入源选择寄存器 ALUOut（Beq指令） |
| | 10 | PC输入源选择转移地址 PC[31:28] IR[25:0]<<2（j指令） |

4.2 多周期控制器设计

❖ ALU控制单元

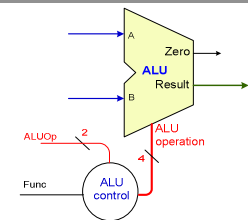
➤ 输入：

- 指令的Func字段（指令5:0位）

- 由主控制单元生成的 **ALUOp**

➤ ALUOp指明ALU的运算类型

- 00：访存指令所需的加法
- 01：beq指令所需的减法
- 10：R型指令功能码字段决定



| 指令 | Func字段 | ALUOp | ALU运算类型 | ALU operation |
|-----|---------|-------|---------|---------------|
| Lw | XXXXXX | 00 | 加 | 0010 |
| Sw | XXXXXX | 00 | 加 | 0010 |
| Beq | XXXXXX | 01 | 减 | 0110 |
| Add | 100 000 | 10 | 加 | 0010 |
| Sub | 100 010 | 10 | 减 | 0110 |
| And | 100 100 | 10 | 与 | 0000 |
| Or | 100 101 | 10 | 或 | 0001 |

4.2 多周期控制器设计

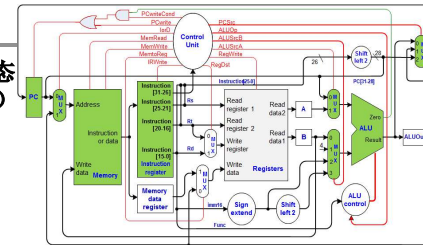
多周期数据通路汇总

| 时钟周期 | R型指令 | Lw指令 | Sw指令 | Beq指令 | J指令 |
|------|-------------------------------------|--|--------------------------|---|--|
| TC1 | | $IR \leftarrow M[PC]$ $PC \leftarrow PC + 4$ | | | |
| TC2 | | $A \leftarrow R[IR[25:21]], B \leftarrow R[IR[20:16]]$ $ALUOut \leftarrow PC + Signext[IR[15:0]] < 2$ | | | |
| TC3 | $ALUOut \leftarrow A \text{ op } B$ | $ALUOut \leftarrow A + Signext[IR[15:0]]$ | | If (A-B=0) then $PC \leftarrow ALUOut$ | $PC \leftarrow PC[31:28] \parallel IR[25:0] < 2$ |
| TC4 | $R[IR[15:11]] \leftarrow ALUOut$ | $DR \leftarrow M[ALUOut]$ | $M[ALUOut] \leftarrow B$ | | |
| TC5 | | $R[IR[20:16]] \leftarrow DR$ | | | |

4.2 多周期控制器设计

1. 数据通路控制信号状态 (P0, 所有指令TC1)

- $IR \leftarrow M[PC]$
- $PC \leftarrow PC + 4$



| 通路部件 | 存储器地址MUX | 存储器 | IR | ALU-A端MUX | ALU-B端MUX | ALU | PC输入端MUX | PC |
|--------|----------|---------|---------|-------------|--------------|------------|------------|---------|
| 操作 | 选择PC | 读出 | 写入 | 选择PC | 常数4 | 加 | 选择ALU | 写入 |
| 控制信号状态 | lorD=0 | MemRead | IRWrite | ALUSrcA = 0 | ALUSrcB = 01 | ALUOp = 00 | PCSrc = 00 | PCWrite |

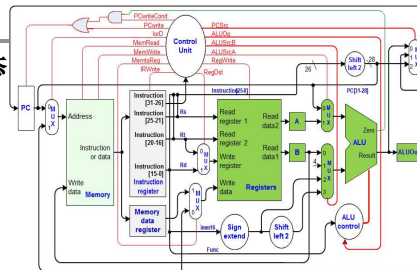
状态ST0:

- MemRead有效
- IRWrite有效
- PCWrite有效
- lorD = 0
- ALUSrcA = 0
- ALUSrcB = 01
- ALUOp = 00
- PCSrc = 00

4.2 多周期控制器设计

2. 数据通路控制信号状态 (P1, 所有指令TC2)

- $A \leftarrow R[IR[25:21]], B \leftarrow R[IR[20:16]]$
- $ALUOut \leftarrow PC + Signext[IR[15:0]] < 2$



| 通路部件 | 寄存器堆 | ALU-A端MUX | ALU-B端MUX | ALU |
|--------|------|-------------|--------------|------------|
| 操作 | 无 | 选择PC | 选择移位器 | 加 |
| 控制信号状态 | 无 | ALUSrcA = 0 | ALUSrcB = 11 | ALUOp = 00 |

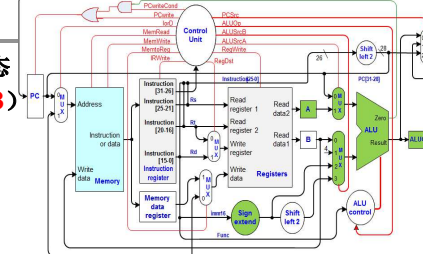
状态ST1:

- ALUSrcA = 0
- ALUSrcB = 11
- ALUOp = 00

4.2 多周期控制器设计

3. 数据通路控制信号状态 (P2, Lw/Sw指令TC3)

- $ALUOut \leftarrow A + Signext[IR[15:0]]$



| 通路部件 | ALU-A端MUX | ALU-B端MUX | ALU |
|--------|-------------|--------------|------------|
| 操作 | 选择寄存器A | 选择符号扩展 | 加 |
| 控制信号状态 | ALUSrcA = 1 | ALUSrcB = 10 | ALUOp = 00 |

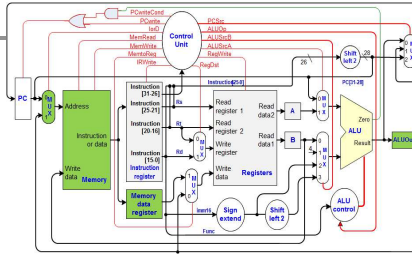
状态ST2:

- ALUSrcA = 1
- ALUSrcB = 10
- ALUOp = 00

4.2 多周期控制器设计

4. 数据通路控制信号状态 (P3, Lw指令TC4)

- ◆ $DR \leftarrow M[ALUOut]$



| 通路部件 | 存储器地址端 MUX | 存储器 | 数据寄存器DR |
|--------|------------|---------|---------|
| 操作 | 选择ALUOut | 读出 | 写入 |
| 控制信号状态 | lorD = 1 | MemRead | 无 |

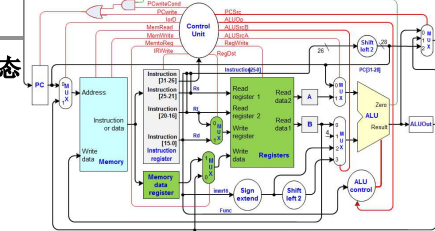
状态ST3:

- lorD = 1
- MemRead有效

4.2 多周期控制器设计

5. 数据通路控制信号状态 (P4, Lw指令TC5)

- ◆ $R[R[20:16]] \leftarrow DR$



| 通路部件 | 寄存器堆写地址MUX | 寄存器堆写数据MUX | 寄存器堆 |
|--------|------------|--------------|----------|
| 操作 | 选择Rt | 选择DR | 写入 |
| 控制信号状态 | RegDst = 0 | MemtoReg = 1 | RegWrite |

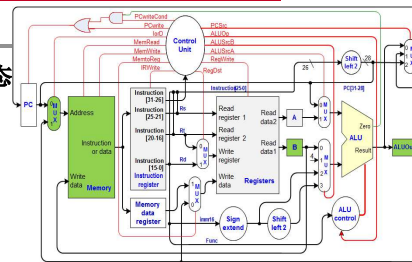
状态ST4:

- RegDst = 0
- MemtoReg = 1
- RegWrite有效

4.2 多周期控制器设计

6. 数据通路控制信号状态 (P5, Sw指令TC4)

- ◆ $M[ALUOut] \leftarrow B$



| 通路部件 | 存储器地址MUX | 存储器 | |
|--------|----------|----------|--|
| 操作 | 选择ALUOut | 写入 | |
| 控制信号状态 | lorD = 1 | MemWrite | |

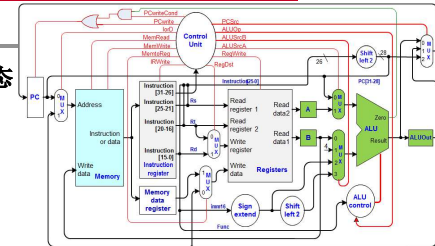
状态ST5:

- lorD = 1
- MemWrite有效

4.2 多周期控制器设计

7. 数据通路控制信号状态 (P6, R型指令TC3)

- ◆ $ALUOut \leftarrow A \text{ op } B$



| 通路部件 | ALU-A端MUX | ALU-B端MUX | ALU |
|--------|-------------|--------------|------------|
| 操作 | 选择寄存器A | 选择寄存器B | 根据Func字段运算 |
| 控制信号状态 | ALUSrcA = 1 | ALUSrcB = 00 | ALUOp = 10 |

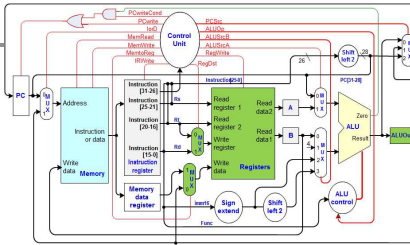
状态ST6:

- ALUSrcA = 1
- ALUSrcB = 00
- ALUOp = 10

4.2 多周期控制器设计

8. 数据通路控制信号状态 (P7, R型指令TC4)

- ◆ $R[IR[15:11]] \leftarrow ALUOut$



| 通路部件 | 寄存器堆写地址MUX | 寄存器堆写数据MUX | 寄存器堆 |
|--------|------------|--------------|----------|
| 操作 | 选择Rd | 选择ALUOut | 写入 |
| 控制信号状态 | RegDst = 1 | MemtoReg = 0 | RegWrite |

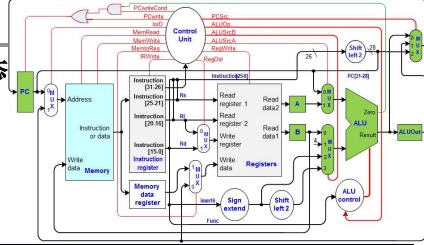
状态ST7:

- RegDst = 1
- MemtoReg = 0
- RegWrite有效

4.2 多周期控制器设计

9. 数据通路控制信号状态 (P8, Beq指令TC3)

- ◆ If (A - B == 0) then
PC ← ALUOut



| 通路部件 | ALU-A端 MUX | ALU-B端 MUX | ALU | PC输入端 MUX | PC |
|--------|-------------|--------------|------------|------------|-------------|
| 操作 | 选择寄存器A | 选择寄存器B | 减 | 选择ALUOut | 有条件写入 |
| 控制信号状态 | ALUSrcA = 1 | ALUSrcB = 00 | ALUOp = 01 | PCSrc = 01 | PCWriteCond |

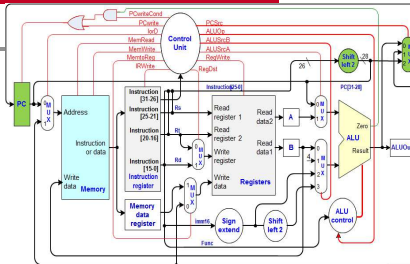
状态ST8:

- ALUSrcA = 1
- ALUSrcB = 00
- ALUOp = 01
- PCSrc = 01
- PCWriteCond有效

4.2 多周期控制器设计

10. 数据通路控制信号状态 (P9, J指令TC3)

- ◆ $PC \leftarrow PC[31:28] \parallel IR[25:0] < 2$

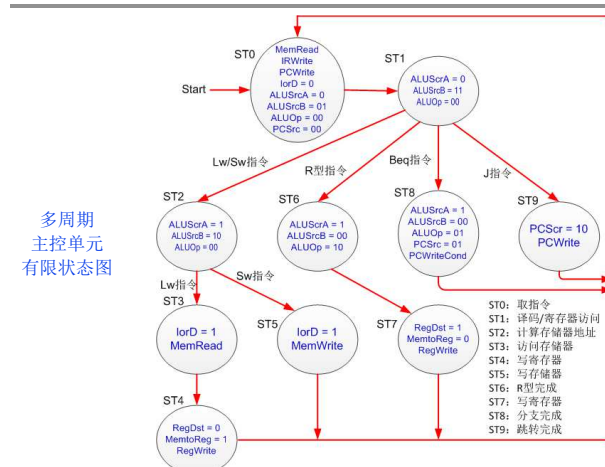


| 通路部件 | PC输入端MUX | PC |
|--------|------------|---------|
| 操作 | 选择转移地址 | 写入 |
| 控制信号状态 | PCSrc = 10 | PCWrite |

状态ST9:

- PCSrc = 10
- PCWrite有效

4.2 多周期控制器设计——主控单元



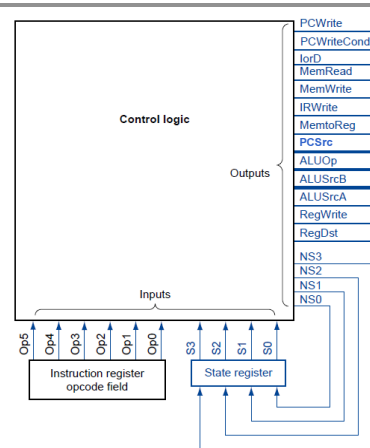
4.2 多周期控制器设计——主控制单元

❖ 有限状态机FSM

- ST0、ST1等10个状态
- 状态编码用4位二进制表示，如 $S_3S_2S_1S_0=0110$ 表示ST6
- 次态用 $NS_3NS_2NS_1NS_0$ 表示

❖ 摩尔型FSM

- 控制信号输出仅取决于当前状态
- 次态由现态和当前输入（IR寄存器中6位操作码）决定



4.2 多周期控制器设计——主控制单元

❖ 控制信号逻辑示例

控制信号仅和当前状态有关

$$PCWrite = ST0 + ST9$$

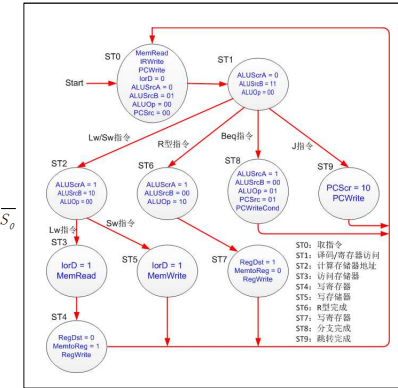
$$= S_3S_2S_1S_0 + S_3S_2S_1S_0$$

$$ALUSrcA = ST2 + ST6 + ST8$$

$$= S_3S_2S_1S_0 + S_3S_2S_1S_0 + S_3S_2S_1S_0$$

$$RegWrite = ST4 + ST7$$

$$= S_3S_2S_1S_0 + S_3S_2S_1S_0$$

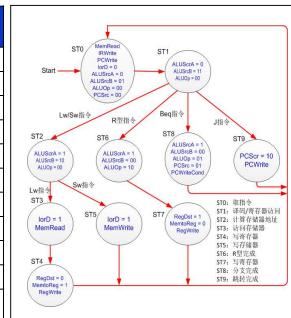


4.2 多周期控制器设计——主控制单元

❖ 次态信号逻辑，次态输出与当前状态和当前输入（Opcode）相关

次态信号真值表

| 当前状态输入 $S_3S_2S_1S_0$ | 操作码输入 ($Op_5Op_4Op_3Op_2Op_1Op_0$) | 次态输出 $NS_3NS_2NS_1NS_0$ |
|--------------------------|---|----------------------------|
| 0000 (ST0) | | 0001 (ST1) |
| 0001 (ST1) | 100011 (lw指令) | 0010 (ST2) |
| 0001 (ST1) | 101011 (sw指令) | 0010 (ST2) |
| 0001 (ST1) | 000000 (R型指令) | 0110 (ST6) |
| 0001 (ST1) | 000100 (Beq指令) | 1000 (ST8) |
| 0001 (ST1) | 000010 (J指令) | 1001 (ST9) |
| 0010 (ST2) | 100011 (lw指令) | 0011 (ST3) |
| 0010 (ST2) | 101011 (sw指令) | 0101 (ST5) |
| 0011 (ST3) | | 0100 (ST4) |
| 0110 (ST6) | | 0111 (ST7) |
| ST4, ST5, ST7, ST8, ST9 | | 0000 (ST0) |



$$NS_3 = ST1(Beq + J)$$

$$= S_3S_2S_1S_0(Op_5Op_4Op_3Op_2Op_1Op_0 + Op_5Op_4Op_3Op_2Op_1Op_0)$$

第六讲 MIPS处理器设计

一. 处理器设计概述

二. MIPS模型机

三. MIPS单周期处理器设计

1. 单周期数据通路设计
2. 单周期控制器设计
3. 单周期性能分析

四. MIPS多周期处理器设计

1. 多周期数据通路设计
2. 多周期控制器设计
3. 多周期性能分析

五. MIPS流水线处理器设计

4.3 多周期性能分析

❖ 假设主要功能单元的操作时间

- 存储器 : 200ps
- ALU : 100ps
- 寄存器堆: 50ps
- 多路复用器、控制单元、PC、符号扩展单元、线路没有延迟

各类指令执行时间

| 步骤 | R型指令 | Lw指令 | Sw指令 | Beq指令 | J指令 | 执行时间 |
|-----------|---|--------------------------------|------------------------------|-------|--------------------------------|-------|
| 取指令 | IR ← M[PC], PC ← PC + 4 | | | | | 200ps |
| 读寄存器/译码 | A ← R[IR[25:21]], B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]] < 2 | | | | | 100ps |
| 计算 | ALUOut ← A op B | ALUOut ← A + Signext[IR[15:0]] | If (A-B==0) then PC ← ALUOut | | PC ← PC[31:28] IR[25:0] < 2 | 100ps |
| R型完成/访问内存 | R[IR[15:11]] ← ALUOut | DR ← M[ALUOut] | M[ALUOut] ← B | | | 200ps |
| 写寄存器 | | R[IR[20:16]] ← DR | | | | 50ps |

4.2 多周期控制器设计

❖ 时钟周期

- 时钟周期取各步骤中最长的时间, 200ps

各类指令执行时间

| 时钟周期 | R型指令 | Lw指令 | Sw指令 | Beq指令 | J指令 | 周期时间 |
|------|---|--------------------------------|------------------------------|-------|--------------------------------|-------|
| TC1 | IR ← M[PC], PC ← PC + 4 | | | | | 200ps |
| TC2 | A ← R[IR[25:21]], B ← R[IR[20:16]] ALUOut ← PC + Signext[IR[15:0]] < 2 | | | | | 200ps |
| TC3 | ALUOut ← A op B | ALUOut ← A + Signext[IR[15:0]] | If (A-B==0) then PC ← ALUOut | | PC ← PC[31:28] IR[25:0] < 2 | 200ps |
| TC4 | R[IR[15:11]] ← ALUOut | DR ← M[ALUOut] | M[ALUOut] ← B | | | 200ps |
| TC5 | | R[IR[20:16]] ← DR | | | | 200ps |

4.3 多周期性能分析

❖ 各型指令所需的时钟周期数和时间

- R型指令 : 800ps
- lw指令 : 1000ps
- sw指令 : 800ps
- beq指令 : 600ps
- j指令 : 600ps

❖ 假设指令在程序中出现的频率

- lw指令 : 25%
- sw指令 : 10%
- R型指令 : 45%
- beq指令 : 15%
- j指令 : 5%

❖ 则一条指令的平均CPI

- $5 \times 25\% + 4 \times 10\% + 4 \times 45\% + 3 \times 15\% + 3 \times 5\% = 4.05$

❖ 一条指令的平均执行时间:

- $1000 \times 25\% + 800 \times 10\% + 800 \times 45\% + 600 \times 15\% + 600 \times 5\% = 810ps$

计算机性能评价

❖ 响应时间与吞吐量

- 响应时间: 从提交作业到完成作业所花费的时间
 - 响应时间是完成一个任务所花的时间总和, 包括内存访问时间、执行IO操作的时间、以及运行必要的操作系统代码所需的时间。
- 吞吐量: 一定时间间隔内完成的作业数
 - 多任务操作系统更侧重于优化系统的整体吞吐量, 而不会特别最小化某个特定程序的响应时间
- 个人用户更关心响应时间, 企业级计算机的管理人员更关心吞吐量
- 对于企业级计算机以外的应用, 响应时间是评价计算机性能的主要依据

计算机性能评价

❖ 响应时间与CPU执行时间

- 对于多任务系统，应该从响应时间中去除因为等待I/O操作而花去的时间和CPU执行其他程序所花费的时间，为此引入**CPU执行时间**的概念。
- CPU执行时间是CPU真正花在运行一个程序上的时间。

程序的CPU执行时间

$$\begin{aligned} &= \text{程序的CPU时钟周期数} \times \text{时钟周期长度} \\ &= \frac{\text{程序的CPU时钟周期数}}{\text{时钟频率}} \end{aligned}$$

程序的CPU时钟周期数

$$= \text{程序的指令数} \times \text{每条指令的平均时钟周期数}$$

计算机性能评价

❖ CPI: 指令平均执行时钟周期数

- CPI: **C**lock **c**ycles **P**er **I**nstruction.
- 不同指令功能不同，所需时间也不同，CPI只是某一机器中一个程序或程序片段每条指令所用时钟周期的平均值。
- 不同指令集的CPI比较没有实际意义。

CPU 时间=CPU 时钟周期数/频率;

CPU 时间=CPU 时钟周期数*时钟周期长;

平均时钟周期数 CPI=CPU 时钟周期数/IC (指令的条数);

CPU 时间= (IC*CPI) /频率 f;

$$\text{CPU 的时钟周期数} = \sum_{i=1}^n (CPI_i \times I_i)$$

$$CPI = \frac{\sum_{i=1}^n (CPI_i \times I_i)}{IC} = \sum_{i=1}^n (CPI_i \times \frac{I_i}{IC})$$

计算机性能评价

❖ MIPS: 百万指令每秒

- MIPS: **M**illion **I**nstruction **P**er **S**econd
- 不同指令集的MIPS比较没有实际意义
- 即使同一台机器，用不同的测试程序测出来的MIPS值也可能不一样。

$$MIPS = \frac{\text{指令条数}}{\text{执行时间} \times 10^6} = \frac{f}{CPI \times 10^6}$$

❖ MFLOPS: 百万浮点数操作每秒

- MFLOPS: **M**illion **F**loating point **O**perations **P**er **S**econd
- 可以比较不同机器的浮点运算能力, 但有局限性
- MFLOPS不仅和机器有关, 也和所用测试程序有关
- MFLOPS与整数、浮点操作的比例有关

$$MFLOPS = \frac{\text{程序中的浮点操作次数}}{\text{执行时间} \times 10^6}$$

计算机性能评价

❖ 影响计算机性能的因素

- **指令数**: 取决于指令集体系结构(ISA), 与指令集的具体实现无关; 编译器对指令数具有很大的影响;
- **CPI**: 机器的实现细节(存储系统结构、处理器结构); 测试程序包含的各类指令的组成等;
- **时钟周期**: 与机器的实现细节密切相关

| 影响因素 | 影响 |
|--------|--------------|
| 算法 | 指令数、CPI |
| 程序设计语言 | 指令数、CPI |
| 编译器 | 指令数、CPI |
| ISA | 指令数、CPI、时钟周期 |
| 硬件实现 | CPI、时钟周期 |

计算机性能评价

❖ 示例一

假设在一台 40MHz 处理机上运行 200,000 条指令的目标代码，程序主要由四种指令组成。根据程序跟踪实验结果，已知指令混合比和每种指令所需的指令数如下。计算在单处理机上用跟踪数据运行程序的平均 CPI，并根据所得的 CPI，计算相应的 MIPS 速率。

| 指令类型 | CPI | 指令混合比 |
|--------------|-----|-------|
| 算术和逻辑 | 1 | 60% |
| 高速缓存命中的加载/存储 | 2 | 18% |
| 转移 | 4 | 12% |
| 高速存储缺失的存储器访问 | 8 | 10% |

[解]

$$CPI = 1 \times 60\% + 2 \times 18\% + 4 \times 12\% + 8 \times 10\% = 2.24$$
$$MIPS = f / (CPI \times 10^6) = (40 \times 10^6) / (2.24 \times 10^6) = 17.86$$

计算机性能评价

❖ 示例二

对于一台 400MHz 计算机执行标准测试程序，程序中指令类型，执行数量和平均时钟周期数如下：

| 指令类型 | 指令执行数量 | 平均时钟周期数 |
|------|--------|---------|
| 整数 | 45000 | 1 |
| 数据传送 | 75000 | 2 |
| 浮点 | 8000 | 4 |
| 分支 | 1500 | 2 |

求该计算机的有效 CPI、MIPS 和程序执行时间。

解：
$$CPI = \sum (IC_i \times CPI_i) / IC$$

$$CPI = \frac{45000 \times 1 + 75000 \times 2 + 8000 \times 4 + 1500 \times 2}{45000 + 75000 + 8000 + 1500} = 1.776$$

$$MIPS \text{ 速率} = \frac{f}{CPI} = \frac{400 \times 10^6}{1.776} = 225.225 MIPS$$

程序执行时间=
$$(45000 \times 1 + 75000 \times 2 + 8000 \times 4 + 1500 \times 2) / (400 \times 10^6) = 5.75 \times 10^{-4} s$$